# Using Variation Propagation for Model-Driven Management of a System Family[*]

Patrick Tessier[1], Sébastien Gérard[1], François Terrier[1], and Jean-Marc Geib[2]

[1] CEA/List Saclay, F-91191 Gif sur Yvette Cedex, France
{Patrick.Tessier, Sebastien.Gerard, Francois.Terrier}@cea.fr
[2] LIFL, Laboratoire d'informatique Fondamentale de Lille, Université des Sciences et Technologies de Lille, 59655 Villeneuve d'Ascq Cedex
Jean-Marc.Geib@lifl.fr

**Abstract.** A system family model (SFM) contains a set of common elements and a set of variable elements known as variation points. Variability modeling is a source of numerous problems: how to express variations, how to ensure the consistency of various views and avoid conflicts. Does the SFM cover all the desired systems? To obtain a specific system, known as "derivation", also known as a product, it is necessary to choose certain variation points from among those included in the SFM model by using a feature model (built during application domain analysis) or a decision model (after SF modelling). The SyF approach presented in this article proposes the "variation point propagation" concept as a means for achieving consistency and dealing with potential conflicts between variations. Under this approach, a decision model, generated from the SFM alone, then enables system family management: analyze coverage of the SF application domain, automate the derivation.

## 1 Introduction

The model-driven approach to system development involves the following key phases (Fig. 1. Process used to build a product):

- *Modeling the system to user requirements*. This phase is iterative and progressive. The system model becomes gradually more detailed and complies, at each successive increment, with the expressed requirements. For the approach described in this article, the language used for modeling is UML.
- *Code generation and compilation*, which provide the final application from its model.

A solution for shortening "time-to-market" for systems sharing certain features is to apply the development principles associated with the system family concept [1, 2]. The idea is to build a single model that factorizes parts of models common to all

[*] This work has been partially supported by the Families European project. Programme, ITEA project ip 02009.

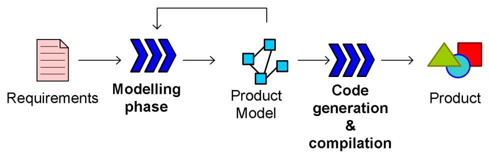

**Fig. 1.** Process used to build a product

systems in a same family and expresses system differences through inclusion of variable elements. This generic model is then known as the SFM or "system family model".

The system family design process usually calls for the following phases (Fig. 2. System family design process) :

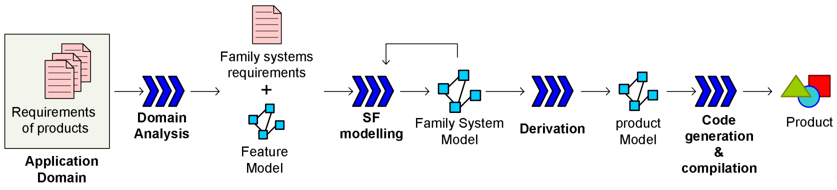

**Fig. 2.** System family design process

- *Domain analysis:* the application domain covers all requirements set for the products. This initial phase, derived from the FODA approach [3], is intended to classify system family requirements and produce a feature diagram. The feature diagram is a tree structure in which each node corresponds to a feature defined as "a prominent or distinctive user-visible aspect, quality or characteristic of a software system or systems" [3]. The characteristics of this tree can be optional, mandatory or linked by an "xor"-type relationship.
- *SF Modeling:* The second phase consists of building the system family model on the basis of these requirements. The SFM contains both variable elements and elements common to all of the desired products. Such modeling is progressive and takes place by increments.
- *Derivation*: Phase three calls for obtaining an application model from the SFM, which serves as its "pattern". To do so, the designer must make a set of choices with regard to the variation points contained in the SFM.
- *Code generation & compilation:* Phase four includes code generation based on the application model derived from the SFM, then compilation of this code to obtain the final product.

For system family designers, joint use of UML and a model-driven approach holds out the promise of enhanced productivity and better quality development, through possible automation of some stages. However, these advantages go hand in hand with certain constraints, which include need for: